PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ^{6} :
G06F 11/00

A1

(11) International Publication Number: WO 00/29950

(43) International Publication Date: 25 May 2000 (25.05.00)

(21) International Application Number: PCT/US99/26899

(22) International Filing Date: 12 November 1999 (12.11.99)

(30) Priority Data:
09/191,701 13 November 1998 (13.11.98) US

(71) Applicant: CHANNELPOINT, INC. [US/US]; Suite 100, 5755 Mark Dabling Boulevard, Colorado Springs, CO 80919 (US).

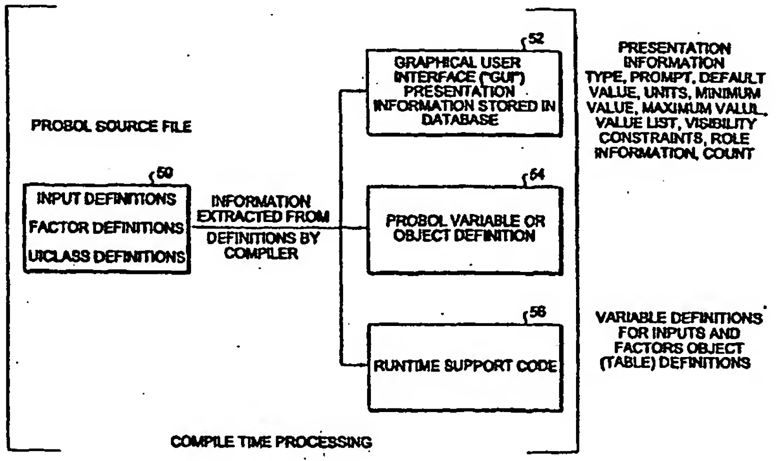
(72) Inventors: LINDLEY, Craig, A.; 6 Sutherland Place, Manitou Springs, CO 80829 (US). JACKSON, Jerry, R.; 11270 Wakely Road, Colorado Springs, CO 80908 (US).

(74) Agents: BURTON, Carol, W. et al.; Hogan & Hartson LLP, Suite 1500, 1200 17th Street, Denver, CO 80202 (US).

(81) Designated States: AU, CA, JP, NZ, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR DYNAMICALLY GENERATING A GRAPHICAL USER INTERFACE FROM COMPUTER PROGRAM SPECIFICATIONS


PROBOL SOURCE FILE
50
INPUT DEFINITIONS
FACTOR DEFINITIONS
UICLASS DEFINITIONS
INFORMATION EXTRACTED FROM DEFINITIONS BY COMPILER
52
GRAPHICAL USER INTERFACE ("GUI") PRESENTATION INFORMATION STORED IN DATABASE
PRESENTATION INFORMATION TYPE, PROMPT, DEFAULT VALUE, UNITS, MINIMUM VALUE, MAXIMUM VALUE, VALUE LIST, VISIBILITY CONSTRAINTS, ROLE INFORMATION, COUNT
54
PROBOL VARIABLE OR OBJECT DEFINITION
56
RUNTIME SUPPORT CODE
VARIABLE DEFINITIONS FOR INPUTS AND FACTORS OBJECT (TABLE) DEFINITIONS
COMPILE TIME PROCESSING

(57) Abstract

As summarized in the figure, this system is a method and computer program product for dynamically generating a graphical user interface from computer program specifications. In a particular implementation of the present invention disclosed herein in conjunction with a computer language called PROBOL™ Inputs, Factors and UIClasses may be utilized in a particular implementation to effectively eliminate the problems previously inherent in the GUI design aspects of a computer program. Programmers using these particular language features may then concentrate on the data storage required for the items requiring data entry and basically, allow the GUI for data entry to generate itself. Since the GUIs are then constructed dynamically, it is no longer possible for the visual representation of the data and the actual data to get out of synchronization with each other thereby alleviating conventional computer program maintenance issues.

AL3

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR DYNAMICALLY GENERATING A GRAPHICAL USER INTERFACE FROM COMPUTER PROGRAM SPECIFICATIONS

Field of the Invention

The present invention relates, in general, to the field of database software and computer program products. More particularly, the present invention relates to a system, method and computer program product for dynamically generating a graphical user interface from computer program specifications of especial utility in implementing rating methodologies for insurance industry applications.

Background of the Invention

Graphic user interfaces ("GUIs") are commonly utilized in conjunction with current computer programs to allow a particular software product to interact with users of the computer programs, such as for data entry and the like. The popularity of these types of user interfaces is demonstrated by their use in conjunction with the Macintosh® (trademark of Apple Computer Corporation, Cupertino CA) computer and the widespread adoption of Windows® 95 and Windows® NT (trademarks of Microsoft Corporation, Redmond, WA) as the operating system of choice on computers with Intel® (trademark of Intel Corporation, Santa Clara, CA) processors.

Unfortunately, writing GUIs into computer programs has long been a time consuming and error prone process. For example, during design of a particular GUI, the programmer must consider at least two aspects of the data entry problem. Firstly, how should the user be presented with the information the program requires for input, and secondly, what data structure should be used to store the data entered by the user and how should it be validated and/or processed. As is apparent, there is generally an extremely tight coupling between the visual presentation of the data and its storage and if one aspect of the two gets out of synchronization with the other, problems occur. Particularly as code maintenance is performed, (i.e. changing the data type of an item) extreme care must be taken by the programmer to ensure that any changes in the data required to be entered in the database are, in turn, reflected in the GUI as required.

At runtime, when a computer program user is actually using the GUI to enter data, the code written by the GUI programmer must: a) display an appropriate screen for the user to interact with which prompts for a particular type of data entry; b) gather the data entered by the user; c) validate the data entered (i.e. ensure the data is of the proper type, the data is within a predetermined range of values, etc.); and d) store and/or act upon the data entered.

Given these facts, the following conclusions can be drawn: a) GUIs are a particularly useful tool to enable users to interact with a computer program; b) traditional GUI development is complex and error prone due to the tight coupling with the non-GUI aspects of the program; c) as a consequence, maintenance of GUI based programs can be difficult; and d) there can actually be more computer programming required in the gathering and validation of data from the user than is used to process the data entered.

SUMMARY OF THE INVENTION

To this end, the present invention advantageously provides a system, method and computer program product for dynamically generating a graphical user interface from computer program specifications. In a particular implementation of the present invention disclosed herein in conjunction with a novel computer language called PROBOL™ developed by Channelpoint, Inc., of Colorado Springs, Colorado, assignee of the present invention, Inputs, Factors and UIClasses may be utilized to effectively eliminate the problems previously inherent in the GUI design aspects of a computer program. Programmers using these particular language features may then concentrate on the data storage required for the items requiring data entry and basically, allow the GUI for data entry to generate itself. Since the GUIs are then constructed dynamically, it is no longer possible for the visual representation of the data and the actual data to get out of synchronization with each other thereby alleviating conventional computer program maintenance issues.

The dynamically generated GUIs produced from the particular Input, Factor and UIClass specifications in the PROBOL language have many common features, including: a) the specification of the prompt to be displayed to the user when data entry is required; b) the ability to specify a minimum value, a maximum value, a minimum and maximum value or a list of values which constrains the data entered by the user (control will be returned to the PROBOL program when all user entered data values satisfy their coded constraints); c) the type of the data to be entered determines the type of user interface control used to gather the user entered data (i.e. if the data is a Boolean type (true/false value only) a checkbox may be used to gather input whereas if the data is numeric or textual, a text box may be presented to the user for data entry. On the other hand, if the user is constrained to choose only from among a list of values, a list box may be presented with these values); d) a special visual-if clause is available which determines (by calling back into PROBOL program code) whether this specific Input, Factor or UIClass should be visible to the user (of course, if the control is not visible to the user, he cannot interact with it and visibility may be controlled by the role the user is currently playing or by date for example); e) a suggested value may be specified for each Input, Factor and/or UIClass attribute to inform the user of the "suggested value" of an item if one exists; f) units may be specified for each Input, Factor and/or UIClass attribute that further clarify what type of data the user is required to input and these hints can control the presentation of the user interface (e.g. if the units are specified as currency, the user interface can further refine its display for gathering monetary values instead of plain numeric values); and g) Factors as implemented have the additional ability to control data entry ranges and range checking on a per role basis (e.g. a user in a more senior role might be accorded the ability to modify a factor more than a user in a less senior role. That is, the senior user might be given lower minimums and higher maximums over which a Factor's value can be varied in a particular implementation of the present invention used for implementing insurance industry rating applications).

As contemplated by the particular embodiment of the system, method and computer program product of the present invention disclosed herein, Inputs, Factors and UIClasses are designed to operate with GUIs. However, because of the modular nature of the computer program code underlying these language features, they might also easily be implemented to interact with a user via a text-based display on a local or a remote computer or through other means.

As utilized in the particular representative embodiment of the present invention disclosed herein in the form of a computer program written utilizing the PROBOL language, the following definitions pertain:

Inputs: Single input items of any simple type including BOOLEAN, STRING, INT (Integer), FLOAT (Floating Point) or DATE. Any number of Inputs can be defined in a PROBOL module and any number of PROBOL modules can make up a methodology. Therefore, a methodology may have numerous Inputs defined that the user will need to be prompted for before a methodology can be run to completion. Inputs may typically be collected before the program in which they are defined is run. Inputs allow the user to interact with a PROBOL program and effectively eliminate the need to write GUIs to gather user inputs and to validate them. They reduce programming effort by removing the GUI design phase of program development and help lower overall program maintenance costs.

Factors: Single items of any simple type including BOOLEAN, STRING, INT, FLOAT or DATE. Factors are like Inputs except that they are given an initial value (calculated within a PROBOL program) which is presented to the user. Factors are collected between runs of a program and give the user the chance to modify the calculated values over specific ranges before the calculations in which the Factor is utilized is run again. Factors allow users to utilize "what-if" types of calculations in a spreadsheet-like manner. The range of values over which a Factor can be manipulated is coded into the Factor's definition in the PROBOL code.

As mentioned previously, ranges may be role specific and Factors allow "fudge" factors to be coded into calculations so that "what-if" scenarios can be accommodated. They allow "what-if" scenarios to be played out but, at the same time, they provide a security framework that will only allow qualified users to see the results. As a consequence, they reduce programming effort, help lower program maintenance costs and provide additional functionality that would otherwise need to be coded separately into the application.

UIClasses: UIClasses are similar in many respects to Inputs but are used for large scale data entry such as for entry of complete rows of database data. For example, a UIClass might describe census data for group members. This data would possibly include first and last names, age, sex and other data relevant to particular individuals. Once a UIClass for census data is defined, the user might then be prompted for all items pertaining to a particular member of the group. Once the GUI (in response to user prompted input) collected all of this data, it would be committed to the database as a single row and the input process would then be repeated for each member of the group.

UIClasses differ from Inputs and Factors in that the data entered by the user is stored in a database instead of being made directly available to the PROBOL program in which the UIClass was defined. UIClasses define not only how the data is to be stored in the database, they also define how the user interface is to be presented and how the user entered data is to be validated. They solve many of the same problems as Inputs but for large amounts of entry data. As noted previously, as the amount of data to be entered by a user increases, the possibility of programming errors in the GUI portion of the program does as well. Just as Inputs serve to reduce programming effort and help lower program maintenance costs as a result of not having to code the GUI portion of a program explicitly, UIClasses further reduce these costs because of the amount of data that can be entered via this particular mechanism.

Particularly disclosed herein is a method and computer program product for dynamically generating a user interface for a computer display. The method comprises the steps of: coding at least one specification in a computer program specifying at least one user input in response thereto; compiling the computer program; executing the compiled computer program; suspending execution of the computer program when the specification is encountered; generating the user interface prompting for the user input; and resuming execution of the computer program upon entry of a last one of the user inputs.

Also disclosed herein is a method and computer program product for use in dynamically generating a user interface for a computer system viewable by a user for data entry. The method comprises the steps of: writing a computer program code module; defining an associated database object for the computer program code module; specifying at least one attribute for the database object; requesting the associated database object in the computer program code module; and compiling the computer program code module. In more specific embodiments, the method may further comprise the steps of: storing the compiled computer program code module in a database; interacting the compiled computer program code module with a computer program runtime code; halting execution of the computer program code module upon encountering the step of requesting the associated database object; extracting information from the associated database object; passing the extracted information to a user interface computer program process; displaying the extracted information to the user; entering data into the computer system in response to the step of displaying the extracted information; and committing the data to the database. In still further embodiments, the method may also further comprise the steps of: validating the entered data prior to the step of committing the data to the database; terminating the step of displaying the extracted information; and resuming the execution of the computer program code module.

Still further disclosed herein is a computer system including a central processing unit, a data entry device and a user viewable display operable in conjunction with the central processing unit. The computer system comprises a compiler operable in conjunction with the central processing unit for compiling a computer program code module, the computer program code module comprising at least one specification specifying at least one user input in response thereto; runtime computer program code operable in conjunction with the central processing unit for executing the compiled computer program code module until the at least one specification is encountered and following entry of a last one of the at least one user input by means of the data entry device; and a user interface computer program process operable in conjunction with the central processing unit for dynamically generating a user interface on the user viewable display prompting a user of the computer system for the at least one user input, the user interface computer program process terminating upon entry of the last one of the at least one user input entered by means of the data entry device.

BRIEF DESCRIPTION OF THE DRAWINGS

The aforementioned and other features and objects of the present invention and the manner of attaining them will become more apparent and the invention itself will be best understood by reference to the following description of a preferred embodiment taken in conjunction with the accompanying drawings, wherein:

Fig. 1 illustrates an exemplary general distributed computing system for possible use in conjunction with the system, method and computer program product of the present invention wherein general purpose computers, workstations or personal computers are connected via communications links of various types;

Fig. 2A is a high level representational view of the compile time processing of Inputs, Factors and UIClasses in accordance with a specific implementation of the present invention in a PROBOL environment;

Fig. 2B is a corresponding, high level representational view of the runtime processing of portions of the compiled information and runtime support code produced in accordance with Fig. 2A;

Fig. 3A is a lower level representational view of the functionality of UIClasses and the operations performed at compile time for UIClass objects by the PROBOL compiler;

Fig. 3B is a corresponding, lower level representational view of the operations performed at runtime for UIClass objects derived from the compiled PROBOL module of Fig. 3A;

Fig. 4A is a logic flow chart corresponding to the compilation process illustrated in Fig. 3A;

Fig. 4B is a logic flow chart corresponding to the runtime process illustrated in Fig. 3B; and

Fig. 5 is a simplified illustration of a representative data entry screen viewable by a user of the system, method and computer program product of the present invention corresponding to the specific example of Figs. 3A and 3B.

DESCRIPTION OF A PREFERRED EMBODIMENT

With reference now to Fig. 1, the environment in which the present invention may be used is illustrated as encompassing a general distributed computing system 10 wherein general purpose computers, workstations or personal computers are connected via communications links of various types, for example a client-server arrangement, an wherein programs and data, many in the form of objects, are made available by various members of the system 10 for execution and access by other members of the system. Some of the elements of a representative computer system 10 are shown including a general purpose workstation 12 and associated server 14 coupled together through an appropriate communications medium 16. The workstation 12 may include input/output ("I/O"), central processing unit ("CPU") and memory sections (not shown) forming a portion of a computer 18 having an associated monitor 20. A keyboard 22 as well as other manual input devices, such as mouse 24, form a portion of the workstation 12 and are coupled to the I/O section of the computer 18 to provide user input data thereto. The monitor 20 is coupled to receive output from the I/O section to provide visually discernible user output. Various types of computer mass storage devices may also be coupled to the I/O section of the computer 18 including a tape drive 26, a hard disk drive 28, a floppy disk drive 30 or a CDROM drive 32 for read/write or read only storage of data. Data may also be communicated to the server 34 by means of the communications medium 16 (such as a local area network "LAN" or wide area network "WAN") intercoupling the I/O portions of the workstation 12 and server 14. The server 14 may comprise, for example, computer tower 34, an associated monitor 36, a keyboard 38 and computer mass storage device 40. The computer program products containing mechanisms to effectuate the apparatus and methods of the present invention may reside in the memory portions of the workstation 12 or server 14 or any of the various associated computer mass storage devices such as tape drive 26, hard disk drive 28, floppy disk drive 30, CDROM drive 32, the communications medium 16, the computer mass storage device 40 or any other device or mechanism for retaining computer program data.

With reference additionally now to Fig 2A, compile time processing for the system, method and computer program product of the present invention is shown. Particularly illustrated is a high level representational view of the compile time processing of Inputs, Factors and UIClasses in accordance with a specific implementation of the present invention in a PROBOL computer program language environment. In this regard, a PROBOL source file 50 may include Input definitions, Factor definitions and UIClass definitions. Information is extracted from the source file 50 by the compiler to develop GUI presentation information 52 (which may be stored in the database), a variable or object definition 54 and runtime support code 56. The GUI presentation information 52 may include, for example, Type, Prompt, Default Value, Units, Minimum Value, Maximum Value, Value List, Visibility Constraints, Role Information, Count and other relevant information. The variable or object definition 54 may include variable definitions for Input and Factor Object definitions.

With reference additionally now to Fig. 2B, runtime processing for the system, method and computer program product of the present invention is shown in the form of a corresponding, high level representational view of the runtime processing of those portions of the compiled information and runtime support code produced in accordance with that shown in preceding figure. In this regard, GUI presentation information 60 (which may comprise, for example, the GUI presentation information 52 of the preceding figure) interacts with the runtime support code 62 (which may comprise, for example, the runtime support code 56 of the preceding figure) to dynamically generate a graphical user interface screen from the aforementioned Input, Factor and/or UIClass computer program specifications on display 64 (for example, monitors 20 or 36 of Fig. 1). The GUI screen prompts the user for data entry which may be effectuated by means of keyboard 66 or other data input mechanism. As shown in block 68, the computer program is further operational to check the user's data entries for compliance with predetermined entry constraints as determined by the GUI presentation information 60 and the input received from the keyboard 66 to then set the PROBOL computer program variable or instantiate an object as shown in block 70.

As described above, it should be noted that the GUI is driven by the data definition directly and that any change to the data definition is immediately reflected in the GUI following compilation of the computer program. Therefore, no separate GUI code needs to be written such that the GUI code can never be out of synchronization with respect to the accompanying program logic.

With reference additionally now to Fig. 3A, a more specific representational view of the functionality of UIClasses in particular, and the operations performed at compile time for UIClass objects by the PROBOL compiler, is shown.

At step 100, a representative PROBOL module is written which defines a UIClass (denominated "Grades" in this example) which will thereafter be used to gather grades for all students in a class. It should be noted that the Grade class definition describes: a) what the data entry screen should be named (i.e. "Grade Input Screen"); b) how the data is to be stored (within the attributes of each Grade object, one of which is created for each student); c) information about each attribute that allows a GUI process to later select the type of control necessary for gathering user inputs; d) a prompt to be given when the user input is requested (e.g. First Name, Last Name and/or Grade) and; e) a range of valid values for the Grade (A, B, C, D or F). Later in the PROBOL computer program code, a REQUEST statement will cause the GUI process to execute at runtime and collect the data for all of the students.

At step 102, the PROBOL compiler is run on the PROBOL code module produced in step 100. The compiler converts the PROBOL source language of the code module into executable code. In the case of a PROBOL module containing UIClasses, the outputs from the compilation include a module of compiled PROBOL code shown in step 106 as well as entries into the associated database at step 104 describing the attributes of the Grades object. The information stored about the Grades object includes the name of the data entry screen, the type of each attribute, the prompts associated with each attribute and any constraints to be placed upon the values of the attributes.

With reference additionally now to Fig. 3B, a corresponding, more specific representational view of the operations performed at runtime for the UIClass objects derived from the compiled PROBOL module 100 of Fig. 3A is shown. The PROBOL module is run at step 110 and it interacts with the PROBOL runtime code as shown in step 112 in the normal course of its execution. At some point during runtime execution, the REQUEST statement is executed which then causes the PROBOL computer program to interact with the Inputs, Factors and UIClasses application program interface ("API") as subsequently shown in step 114. While such a REQUEST statement is being processed, the PROBOL computer program code halts execution.

The Inputs, Factors and UIClasses API extracts information from the associated database 116 about the data to be gathered and passes it to the GUI process at step 118 which then formats it for display, for example, on monitor 120. As data is entered by the user, for example on keyboard 122, it is validated by the API at step 114. If the data is deemed valid, it is committed to the database 116 in the form of Grade objects. On the other hand, if the data is not valid, the GUI process at step 118 flags the error and the user so that the data entry can be corrected. When all of the data has been gathered, the GUI process at step 118 terminates and control returns to the PROBOL program to the statement after the particular UIClass request. Execution of the PROBOL computer program then continues. It should be noted that the GUI process knows nothing about the data it is requested to collect and, in fact, only knows how to collect it and how to validate it.

With reference additionally now to Fig. 4A, a logic flow chart corresponding to the compilation process previously illustrated and described with respect to Fig. 3A is shown. A representative compilation process 130 begins with the initial writing of a PROBOL module defining, for example, UIClass Grades at step 132. The PROBOL module containing the UIClass Grades is compiled at step 134 and the name of the data entry screen, the type of attribute, prompts and any constraints on the values of the attributes are stored in the database at step 136 upon completion of the compilation process.

With reference additionally now to Fig. 4B, a logic flow chart corresponding to the runtime process previously illustrated and described with respect to Fig. 3B is shown. A representative runtime process 150 begins with the execution of the compiled PROBOL module at step 152. As previously described, the module interacts with the PROBOL runtime code at step 154 and continues until a corresponding REQUEST statement is executed at decision step 156. At this point, step 158 halts execution of the code and the runtime code interacts with the Inputs, Factors and UIClasses API at step 160. At step 162, the Inputs, Factors and UIClasses API extracts information from the database which is then passed to the GUI process at step 164.

The GUI process formats the information for display and the user is prompted for data input at step 166. As it is entered, the user entered data is validated by the appropriate API at step 168. If the data is determined to be valid at decision step 170, it is committed to the database in the form of Grade objects. Alternatively, if at decision step 170 the user entered data is determined to be invalid, the error is flagged to the user at step 174 and he is prompted for re-entry of the requisite data. At decision step 176, if the required data entry session has been completed, the GUI process is terminated at step 178 and control is returned to the computer program at a point following the UIClass request at step 180.

With reference additionally now to Fig. 5, a simplified illustration of a representative data entry screen 200 viewable by a user of the system, method and computer program product of the present invention corresponding to the specific example of Figs. 3A, 3B, 4A and 4B is shown. The data entry screen which may be dynamically generated by the system, method and computer program product of the present invention in accordance with the foregoing representative example includes a title field "Grade Input Screen" 202, a "Last Name:" field designator 204 and corresponding data entry field 206, a "First Name:" field designator 208 and corresponding data entry field 210 and a "Grade:" field designator 212 and corresponding data entry field 214.

The data entry field 214 also includes an associated list box 216 which lists acceptable values for user entry of data in the data entry field 214. The data entry screen 200 may also include a "Next" designator 218 and "Done" designator 220 which may be actuatable by the user to signify to the computer program when data entry corresponding to a next individual is to be attempted or the final individual has just been entered respectively.

While there have been described above the principles of the present invention in conjunction with a specific computer program code example and a particular computer programming language, it is to be clearly understood that the foregoing description is made only by way of example and not as a limitation to the scope of the invention. Particularly, it is recognized that the teachings of the foregoing disclosure will suggest other modifications to those persons skilled in the relevant art. Such modifications may involve other features which are already known per se and which may be used instead of or in addition to features already described herein. Although claims have been formulated in this application to particular combinations of features, it should be understood that the scope of the disclosure herein also includes any novel feature or any novel combination of features disclosed either explicitly or implicitly or any generalization or modification thereof which would be apparent to persons skilled in the relevant art, whether or not such relates to the same invention as presently claimed in any claim and whether or not it mitigates any or all of the same technical problems as confronted by the present invention. The applicants hereby reserve the right to formulate new claims to such features and/or combinations of such features during the prosecution of the present application or of any further application derived therefrom.

What is claimed is:

CLAIMS:

1. A method for use in dynamically generating a user interface for a computer system viewable by a user for data entry, said method comprising:

writing a computer program code module;

defining an associated database object for said computer program code module;

specifying at least one attribute for said database object;

requesting said associated database object in said computer program code module; and

compiling said computer program code module.

2. The method of claim 1 further comprising the step of:

storing said compiled computer program code module in a database.

3. The method of claim 2 further comprising the step of:

interacting said compiled computer program code module with a computer program runtime code.

4. The method of claim 3 further comprising the step of:

halting execution of said computer program code module upon encountering said step of requesting said associated database object.

5. The method of claim 4 further comprising the step of:

extracting information from said associated database object.

6. The method of claim 5 further comprising the steps of:

passing said extracted information to a user interface computer program process; and

displaying said extracted information to said user.

7. The method of claim 6 further comprising the steps of:

entering data into said computer system in response to said step of displaying said extracted information; and

committing said data to said database.

8. The method of claim 7 further comprising the step of:

validating said entered data prior to said step of committing said data to said database.

9. The method of claim 8 further comprising the steps of:

terminating said step of displaying said extracted information; and

resuming said execution of said computer program code module.

10. A method for dynamically generating a user interface for a computer display comprising the steps of:

coding at least one specification in a computer program specifying at least one user input in response thereto;

compiling said computer program;

executing said compiled computer program;

suspending execution of said computer program when said at least one specification is encountered;

generating said user interface prompting for said at least one user input; and

resuming execution of said computer program upon entry of a last one of said at least one user input.

11. The method of claim 10 further comprising the step of:

storing said compiled computer program code module in a database.

12. The method of claim 11 further comprising the steps of:

entering data into said computer system in response to said step of generating said user interface; and committing said data to said database.

13. The method of claim 12 further comprising the step of:

validating said entered data prior to said step of committing said data to said database.

14. The method of claim 8 further comprising the steps of:

terminating said step of generating said user interface prior to said step of resuming execution of said computer program.

15. A computer program product comprising:

a computer usable medium having computer readable code embodied therein for dynamically generating a user interface for a computer system viewable by a user for data entry in response to a computer program code module having a defined associated database object and at least one attribute specified therefor, the computer program product comprising:

computer readable program code devices configured to cause a computer to effect requesting said associated database object in said computer program code module; and; and

computer readable program code devices configured to cause a computer to effect compiling said computer program code module.

16. The computer program product of claim 15 further comprising computer readable program code devices configured to cause a computer to effect storing said compiled computer program code module in a database.

17. The computer program product of claim 16 further comprising computer readable program code devices configured to cause a computer to effect interacting said compiled computer program code module with a computer program runtime code.

18. The computer program product of claim 17 further comprising computer readable program code devices configured to cause a computer to effect halting execution of said computer program code module upon encountering said request for said associated database object.

19. The computer program product of claim 18 further comprising computer readable program code devices configured to cause a computer to effect extracting information from said associated database object.

20. The computer program product of claim 19 further comprising computer readable program code devices configured to cause a computer to effect passing said extracted information to a user interface computer program process and displaying said extracted information to said user.

21. The computer program product of claim 20 further comprising computer readable program code devices configured to cause a computer to effect the receipt of data entered into said computer system in response to said step of displaying said extracted information; and committing said data to said database.

22. The computer program product of claim 21 further comprising computer readable program code devices configured to cause a computer to effect validating said entered data prior to committing said data to said database.

23. The computer program product of claim 22 further comprising computer readable program code devices configured to cause a computer to effect terminating display of said extracted information and resuming said execution of said computer program code module.

24. A computer program product comprising:

a computer usable medium having computer readable code embodied therein for dynamically generating a user interface for a computer system display, said computer program product allowing for the coding of at least one specification specifying at least one user input in response thereto, the computer program product comprising:

computer readable program code devices configured to cause a computer to effect compiling said computer program;

computer readable program code devices configured to cause a computer to effect executing said compiled computer program;

computer readable program code devices configured to cause a computer to effect suspending execution of said computer program when said at least one specification is encountered;

computer readable program code devices configured to cause a computer to effect generating said user interface prompting for said at least one user input; and

computer readable program code devices configured to cause a computer to effect resuming execution of said computer program upon entry of a last one of said at least one user input.

25. The computer program product of claim 24 further comprising:

computer readable program code devices configured to cause a computer to effect storing said compiled computer program code module in a database.

26. The computer program product of claim 25 further comprising:

computer readable program code devices configured to cause a computer to allow entering data into said computer system in response to said step of generating said user interface.

27. The computer program product of claim 26 further comprising:

computer readable program code devices configured to cause a computer to effect committing said data to said database.

28. The computer program product of claim 27 further comprising:
computer readable program code devices configured to cause a computer to effect validating said entered data prior to said step of committing said data to said database.

29. The computer program product of claim 28 further comprising:
computer readable program code devices configured to cause a computer to effect terminating said step of generating said user interface prior to said step of resuming execution of said computer program.

30. A computer system including a central processing unit, a data entry device and a user viewable display operable in conjunction with said central processing unit, said computer system comprising:
a compiler operable in conjunction with said central processing unit for compiling a computer program code module, said computer program code module comprising at least one specification specifying at least one user input in response thereto;
runtime computer program code operable in conjunction with said central processing unit for executing said compiled computer program code module until said at least one specification is encountered and following entry of a last one of said at least one user input by means of said data entry device; and
a user interface computer program process operable in conjunction with said central processing unit for dynamically generating a user interface on said user viewable display prompting a user of said computer system for said at least one user input, said user interface computer program process terminating upon entry of said last one of said at least one user input entered by means of said data entry device.

31. The computer system of claim 30 further comprising:

a database for storing said compiled computer program code module.

32. The computer system of claim 30 further comprising:

a user input validator for assessing said at least one user input and flagging invalid input to said user on said user interface display.

33. The computer system of claim 30 further comprising:

a database for storing data corresponding to said at least one user input in the form of a data object.

34. The computer system of claim 30 wherein said user interface comprises at least one data entry field corresponding to said at least one user input.

35. The computer system of claim 34 wherein said user interface further comprises at least one field designator denominating said at least one data entry field.

36. The computer system of claim 34 wherein said user interface comprises a list box associated with said at least one data entry field for indicating to said user possible valid data to be entered in said at least one data entry field.

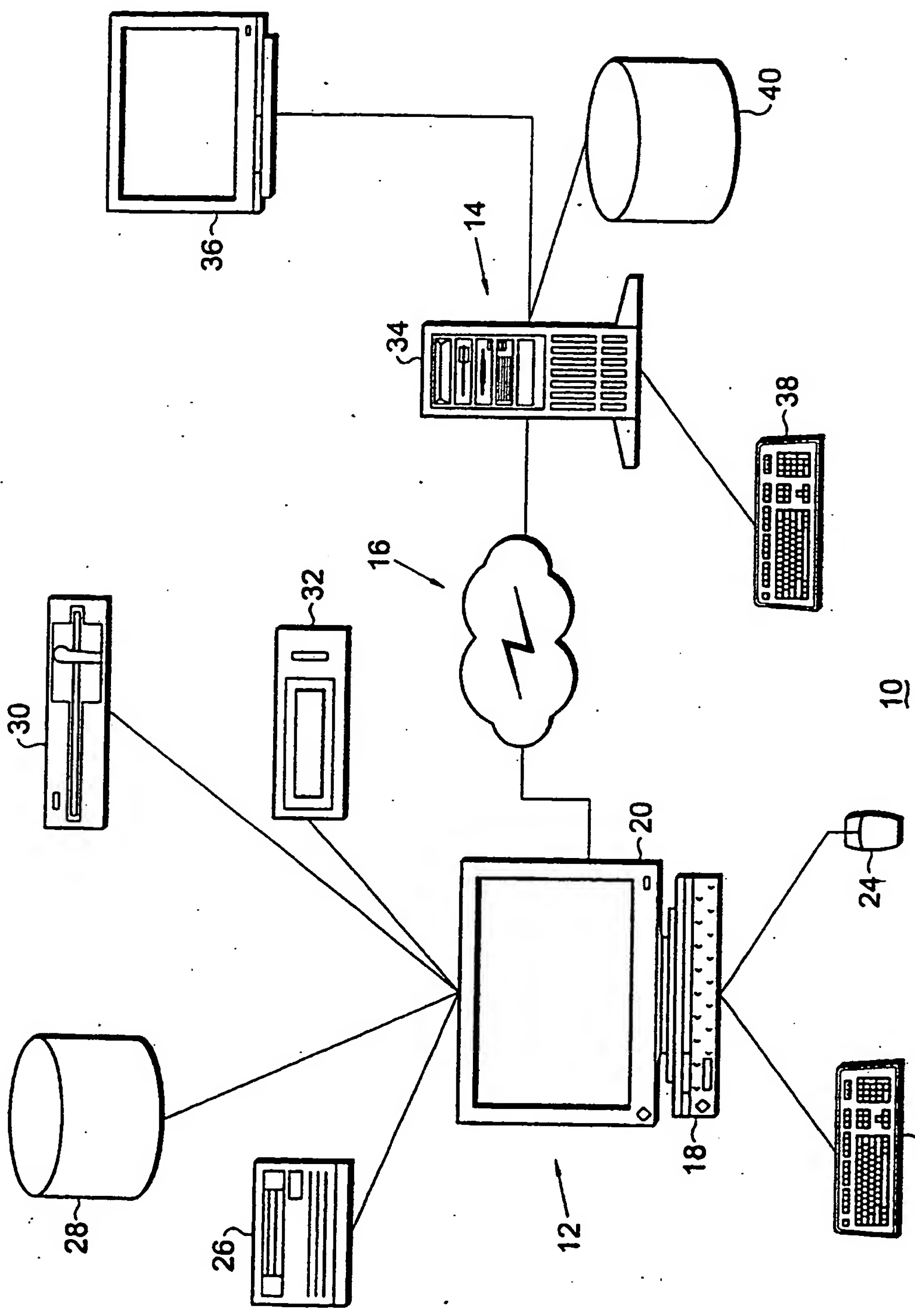


FIG. 1

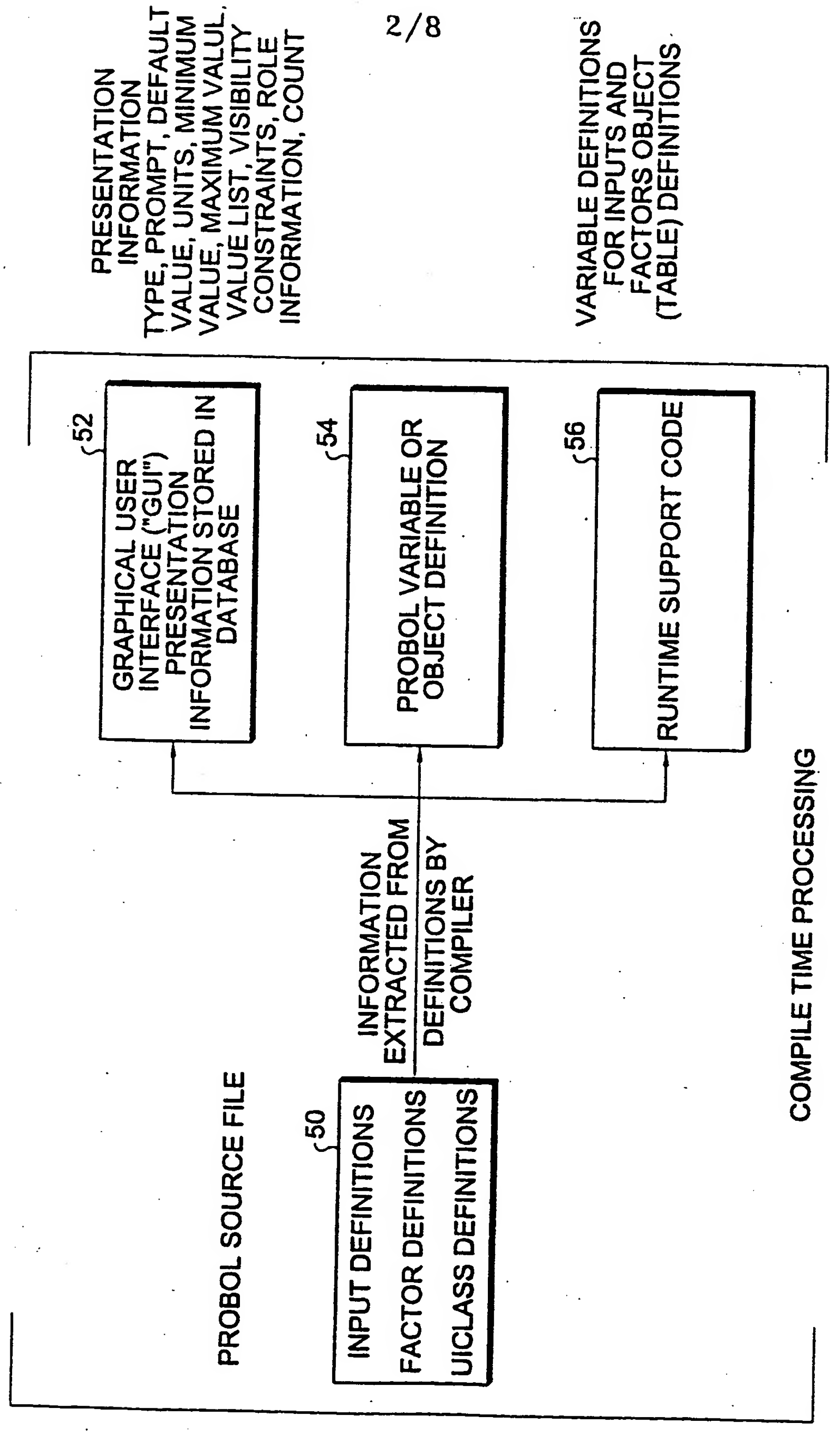

PROBOL SOURCE FILE
50
INPUT DEFINITIONS
FACTOR DEFINITIONS
UICLASS DEFINITIONS
INFORMATION EXTRACTED FROM DEFINITIONS BY COMPILER
52
GRAPHICAL USER INTERFACE ("GUI") PRESENTATION INFORMATION STORED IN DATABASE
54
PROBOL VARIABLE OR OBJECT DEFINITION
56
RUNTIME SUPPORT CODE
PRESENTATION INFORMATION TYPE, PROMPT, DEFAULT VALUE, UNITS, MINIMUM VALUE, MAXIMUM VALUL, VALUE LIST, VISIBILITY CONSTRAINTS, ROLE INFORMATION, COUNT
VARIABLE DEFINITIONS FOR INPUTS AND FACTORS OBJECT (TABLE) DEFINITIONS
COMPILE TIME PROCESSING

FIG. 2A

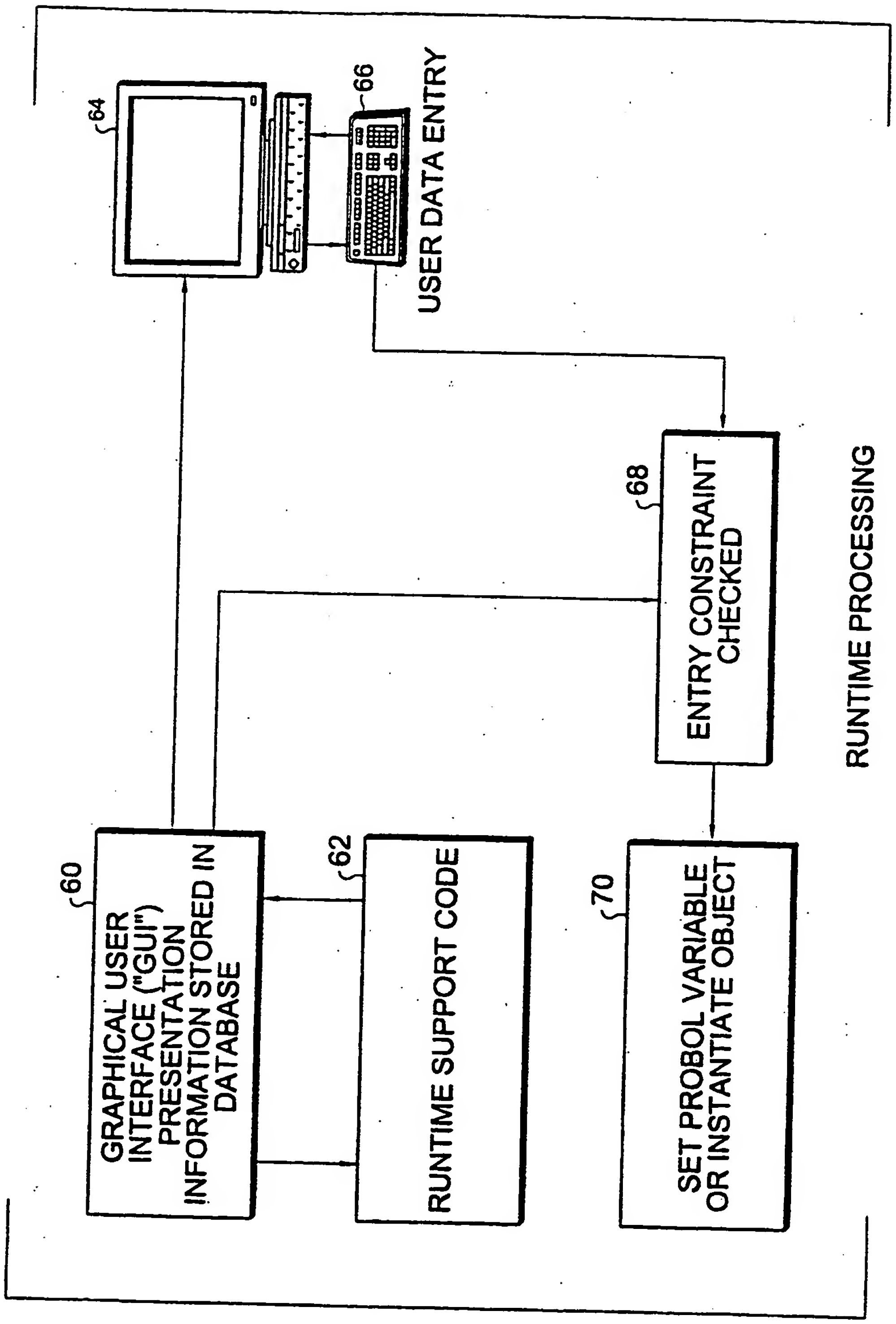
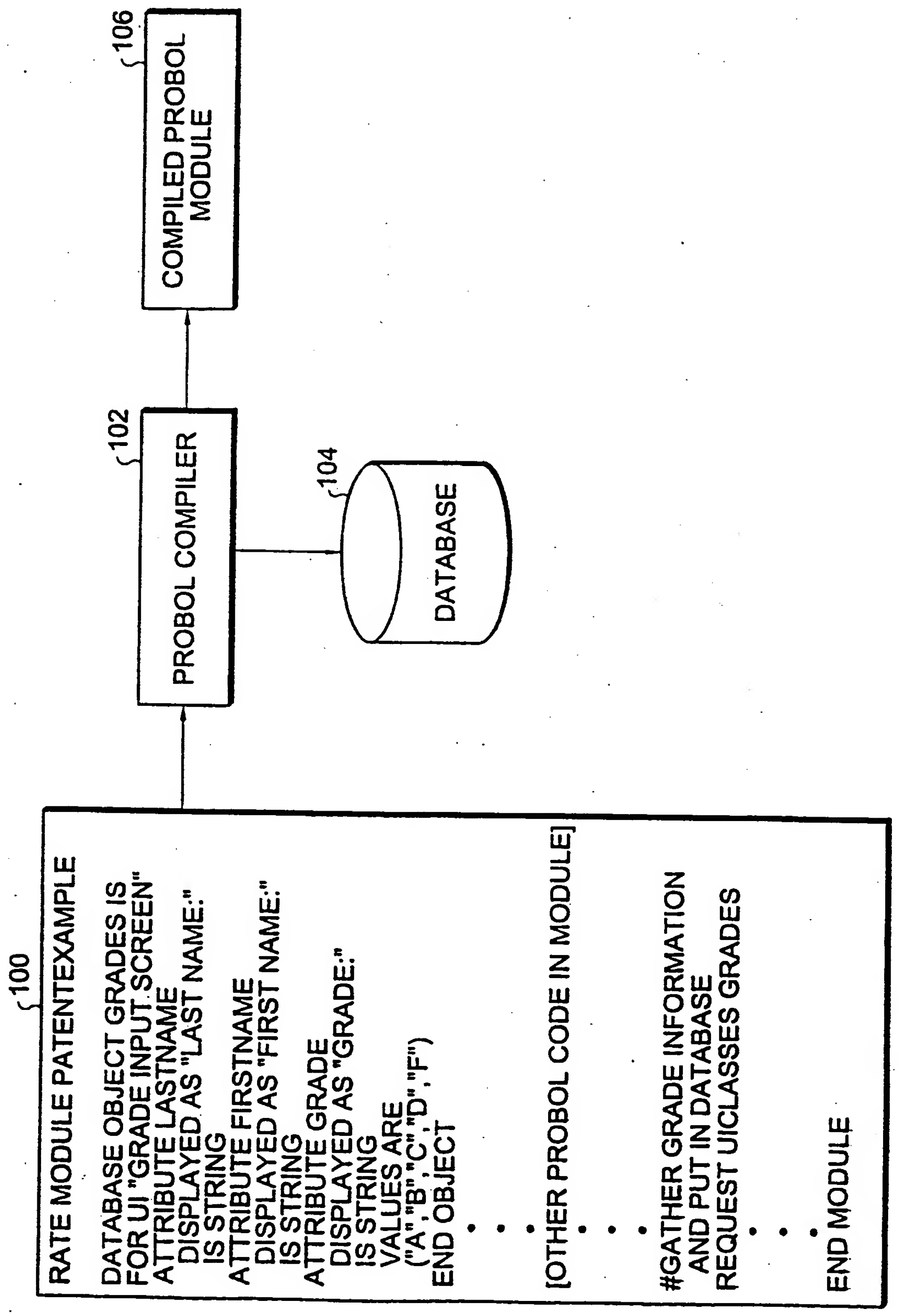

64
66
USER DATA ENTRY
68
ENTRY CONSTRAINT CHECKED
RUNTIME PROCESSING
60
GRAPHICAL USER INTERFACE ("GUI") PRESENTATION INFORMATION STORED IN DATABASE
62
RUNTIME SUPPORT CODE
70
SET PROBOL VARIABLE OR INSTANTIATE OBJECT

FIG. 2B



```
RATE MODULE PATENTEXAMPLE

DATABASE OBJECT GRADES IS
FOR UI "GRADE INPUT SCREEN"
ATTRIBUTE LASTNAME
   DISPLAYED AS "LAST NAME:"
  IS STRING
ATTRIBUTE FIRSTNAME
   DISPLAYED AS "FIRST NAME:"
  IS STRING
ATTRIBUTE GRADE
   DISPLAYED AS "GRADE:"
  IS STRING
  VALUES ARE
   ("A","B","C","D","F")
END OBJECT
   . . .

[OTHER PROBOL CODE IN MODULE]
   . . .

#GATHER GRADE INFORMATION
 AND PUT IN DATABASE
 REQUEST UICLASSES GRADES
   . . .

END MODULE
```

FIG. 3A

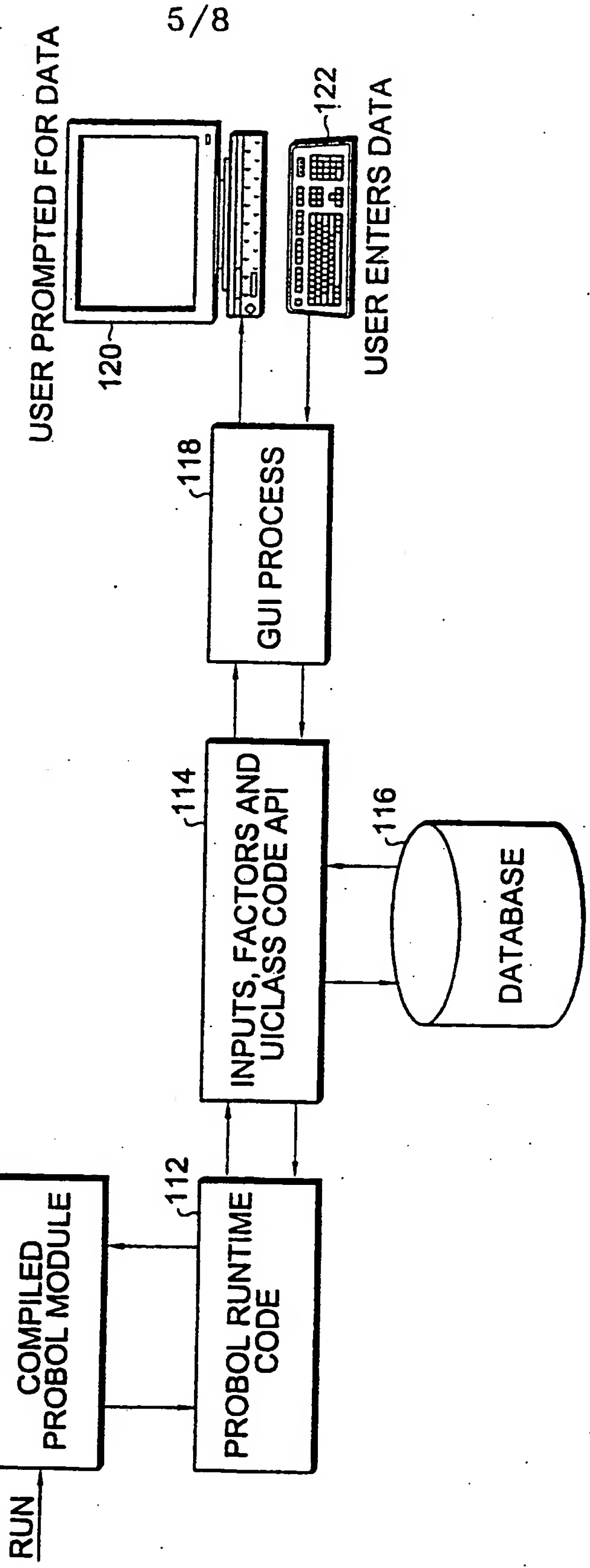

RUN
COMPILED PROBOL MODULE
110
PROBOL RUNTIME CODE
112
INPUTS, FACTORS AND UICLASS CODE API
114
DATABASE
116
GUI PROCESS
118
USER PROMPTED FOR DATA
120
122
USER ENTERS DATA

FIG. 3B

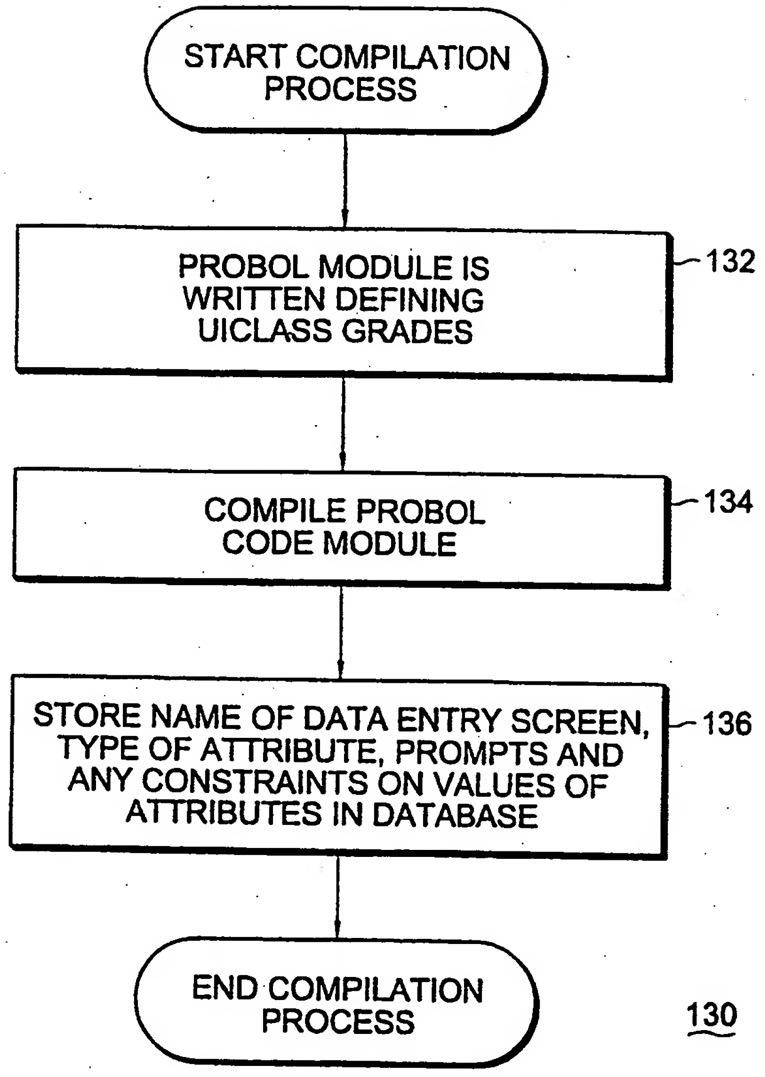

START COMPILATION PROCESS
PROBOL MODULE IS WRITTEN DEFINING UICLASS GRADES
132
COMPILE PROBOL CODE MODULE
134
STORE NAME OF DATA ENTRY SCREEN, TYPE OF ATTRIBUTE, PROMPTS AND ANY CONSTRAINTS ON VALUES OF ATTRIBUTES IN DATABASE
136
END COMPILATION PROCESS
130

FIG. 4A

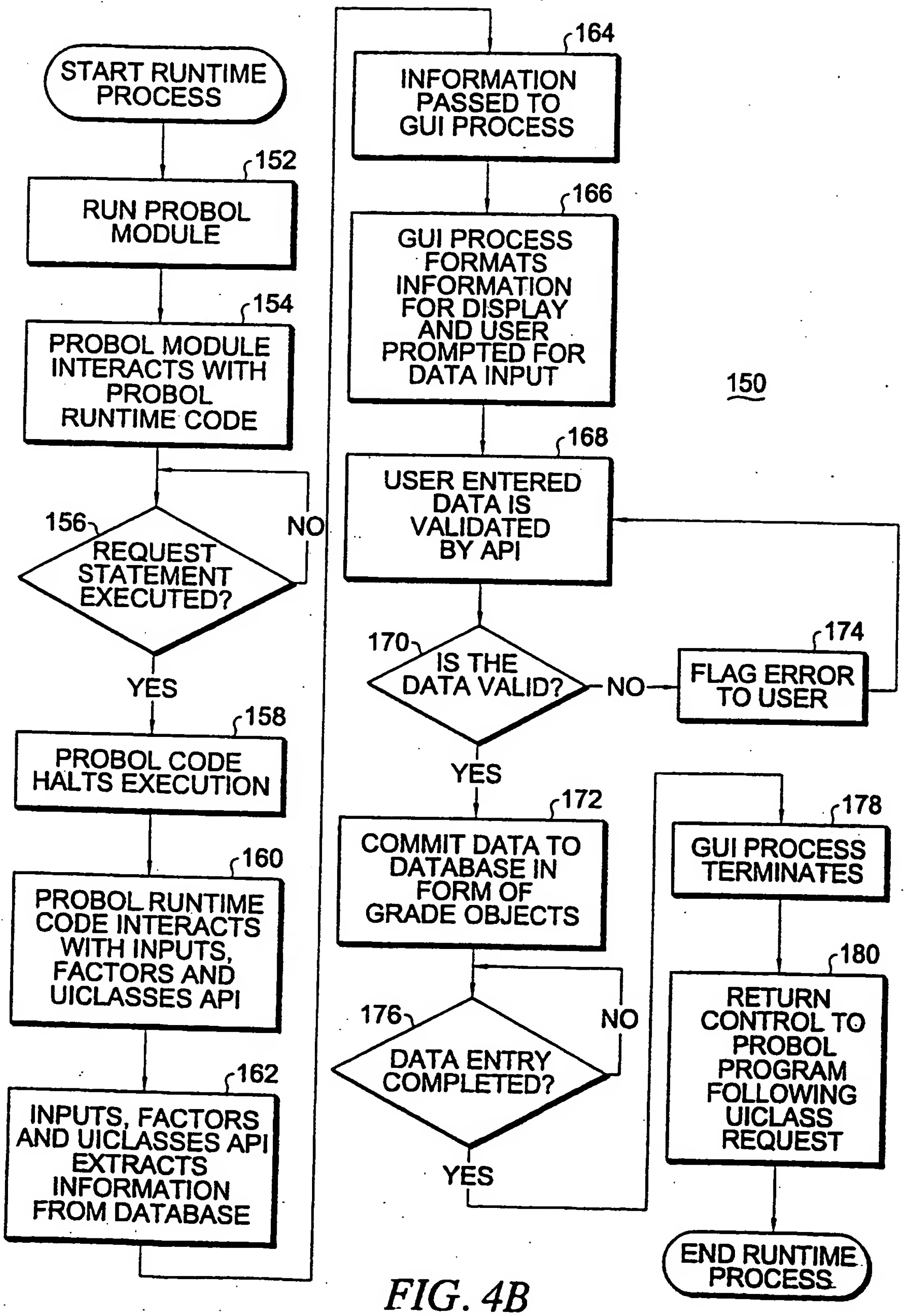

START RUNTIME PROCESS
152
RUN PROBOL MODULE
154
PROBOL MODULE INTERACTS WITH PROBOL RUNTIME CODE
156
REQUEST STATEMENT EXECUTED?
NO
YES
158
PROBOL CODE HALTS EXECUTION
160
PROBOL RUNTIME CODE INTERACTS WITH INPUTS, FACTORS AND UICLASSES API
162
INPUTS, FACTORS AND UICLASSES API EXTRACTS INFORMATION FROM DATABASE
164
INFORMATION PASSED TO GUI PROCESS
166
GUI PROCESS FORMATS INFORMATION FOR DISPLAY AND USER PROMPTED FOR DATA INPUT
150
168
USER ENTERED DATA IS VALIDATED BY API
170
IS THE DATA VALID?
NO
174
FLAG ERROR TO USER
YES
172
COMMIT DATA TO DATABASE IN FORM OF GRADE OBJECTS
176
DATA ENTRY COMPLETED?
NO
YES
178
GUI PROCESS TERMINATES
180
RETURN CONTROL TO PROBOL PROGRAM FOLLOWING UICLASS REQUEST
END RUNTIME PROCESS

FIG. 4B

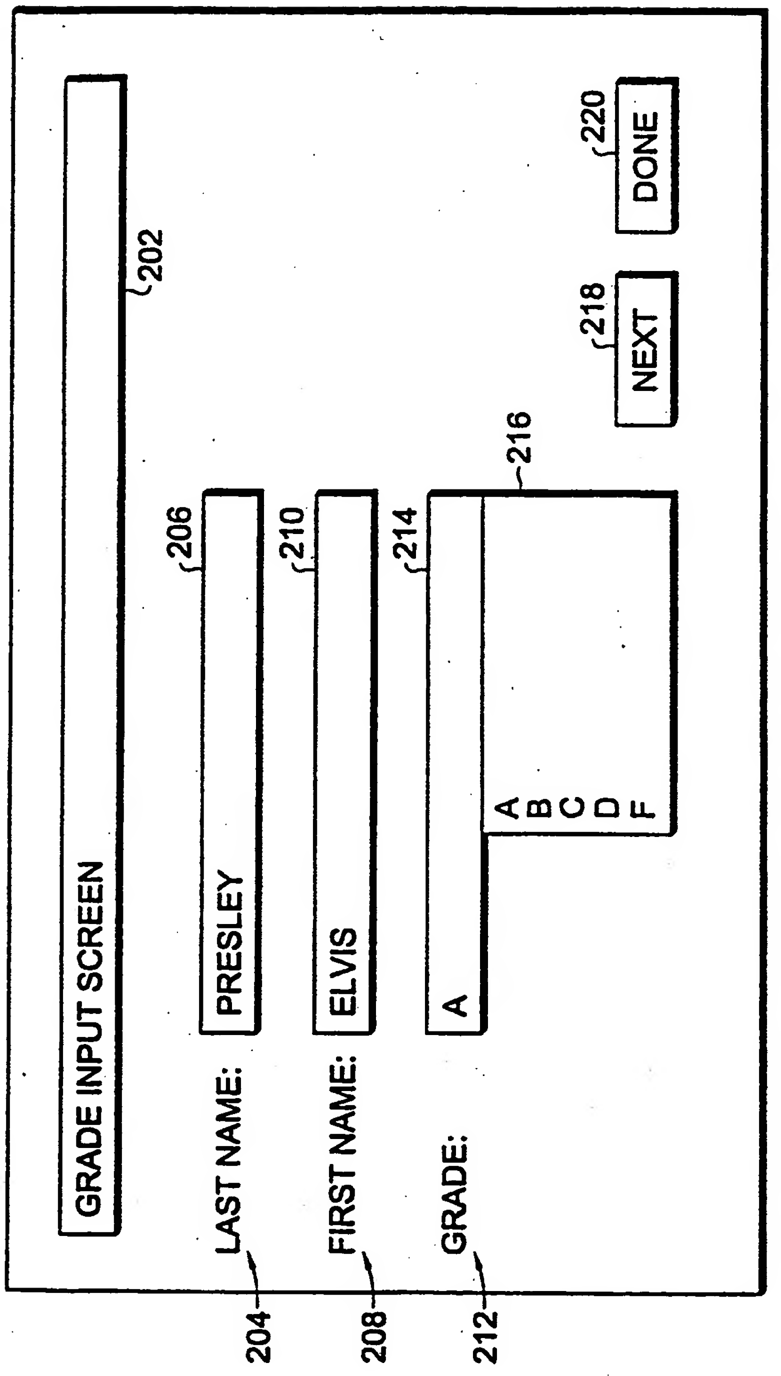

GRADE INPUT SCREEN
202
LAST NAME:
204
PRESLEY
206
FIRST NAME:
208
ELVIS
210
GRADE:
212
A
214
A
B
C
D
F
216
NEXT
218
DONE
220
200

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/26899

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) :G^6F 1100
US CL : 345/326
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED
Minimum documentation searched (classification system followed by classification symbols)
U.S. : 345/326, 333-335, 339, 967; 707/103; 395/701, 702, 704

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y	Visual Basic 5.0 Introduction Student Manual, Ziff-Davis Education, (c)1997, pp. 1-6, 4-1 to 4-8, and 8-13 to 8-15 including fig. 8-2.	1-31 and 33-36 32
Y,E	US 5,995,958 A (XU et al) 30 November 1999, fig. 8-9, col. 9, lines 35 - 67; col. 10, lines 1 - 30.	32
A	BORNING ET AL., "Constraint-Based Tools for Building User Interfaces", ACM Trasnsactions on Graphics, vol. 5, no. 4, October 18, 1986, pp. 345-374.	1-36
A,E	US 6,003,143 A (KIM et al.) 14 December 1999, ALL	1-36

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 01 FEBRUARY 2000	Date of mailing of the international search report 15 FEB. 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer [illegible] MATT KIM Telephone No. (703) 305-9600

Form PCT/ISA/210 (second sheet)(July 1992) ★

[illegible]